

Multimédia FotoPrint

Multimédia

■ Elementos Media

Tags	Descrição
<canvas>	Etiqueta que recorre à utilização de <i>scripts</i> para criar animações e desenhar gráficos vetoriais
<audio>	Define conteúdo relativo a áudio digital
<video>	Define conteúdo relativo a vídeo digital podendo conter áudio digital
<source>	Permite definir vários media para os elementos <video> e <áudio>
<embed>	Define um ponto de ligação para uma aplicação externa (e.g., FLASH)
<track>	Define <i>tracks</i> de texto para elementos media

Áudio, Vídeo e Câmera Web

Multimédia: Áudio

■ Áudio no Browser

```
<html>
  <body>
    <audio>
      <source src="audio/03_Muda_de_vida.ogg" type="audio/ogg">
      <source src="audio/03_Muda_de_vida.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
    </audio>
  </body>
</html>
```

Formatos áudio – Porquê dois ficheiros????

Multimédia: Áudio - Atributos

- “controls”, “autoplay” e “loop”

```
<html>
  <body>
    <audio autoplay controls loop>
      <source src="audio/03_Muda_de_vida.ogg" type="audio/ogg">
      <source src="audio/03_Muda_de_vida.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
  </audio>
  </body>
</html>
```



Para adicionar botões de controlo do áudio
“PLAY”,
“PAUSE”
“VOLUME”

Multimédia: Formatos de Áudio

■ Formatos de Áudio

■ MP3

- Ficheiros de áudio do MPEG
- Mais populares nos *players*
- Combinam boa compressão com elevada qualidade

■ OGG

- Desenvolvido pela fundação Xiph.Org

■ WAV

- Desenvolvido pela IBM e Microsoft.
- Funciona bem no Windows, Macintosh e Linux



Multimédia: Áudio - *Browsers*

■ Formatos de Áudio Suportados pelos *Browsers*

<i>Browser</i>	<i>MP3</i>	<i>Wav</i>	<i>Ogg</i>
Edge/Internet Explorer	Sim	Não	Não
Chrome	Sim	Sim	Sim
Firefox	Sim	Sim	Sim
Safari	Sim	Sim	Não
Opera	Sim	Sim	Sim

Multimédia: Vídeo

■ Vídeo

```
<html>
  <body>
    <video width="320" height="240" controls>
      <source src="video/13_MissionImpossible_KyleCooper.ogv" type="video/ogg">
      <source src="video/13_MissionImpossible_KyleCooper.mp4" type="video/mp4">
    Your browser does not support the video tag.
  </video>
  </body>
</html>
```

Estes dois atributos devem ser colocados para que o *browser* conheça as dimensões do vídeo. Desta forma evita-se alterações na página (efeito de *flicker*)



Multimédia: Formatos de Vídeo

■ Formatos de Vídeo

■ MP4

- Ficheiros MPEG4 com codificador de vídeo H264 e o codificador de áudio AAC
- Utilizado na câmaras de vídeo e TVs mais recentes
- Recomendado pelo YouTube

■ WebM

- Desenvolvido por Mozilla, Opera, Adobe e Google
- Ficheiros WebM com codificador de vídeo VP8 video e codificador de áudio Vorbis

■ Ogg

- Desenvolvido pela fundação Xiph.Org
 - Ficheiros Ogg com codificador de vídeo Theora e codificador de áudio Vorbis
-

Multimédia: Vídeo - *Browsers*

■ Formatos de Vídeo Suportados nos *Browsers*

<i>Browser</i>	<i>MP4</i>	<i>WebM</i>	<i>Ogg</i>
Internet Explorer	Sim	Não	Não
Chrome	Sim	Sim	Sim
Firefox	Sim	Sim	Sim
Safari	Sim	Não	Não
Opera 25	Sim	Sim	Sim

Multimédia: Áudio e Vídeo - Atributos

■ Atributos

Atributos	Descrição
currentSrc	Retorna o URL do vídeo/áudio atual
currentTime	Retorna ou define a posição atual para reprodução do vídeo/áudio
duration	Retorna a duração do vídeo/áudio em segundos
ended	Retorna se a reprodução já terminou ou não
muted	Retorna ou define se o vídeo/áudio fica <i>muted</i> ou não
paused	Retorna se o vídeo/áudio está parado ou não
volume	Retorna ou define o volume

Multimédia: Vídeo - Câmara Web

```

<body>
<h1>Simple web camera display demo</h1>
<video id="video" width="640" height="480" autoplay></video>

<script>
    window.addEventListener("DOMContentLoaded", WebCameraVideo, false);
    function WebCameraVideo () {
        let video = document.getElementById("video");
        let videoObj = {"video": true};

        let errBack = function (error) {
            console.log("Video capture error: ", error.code);
        };

        let getStream = function (stream) {
            video.srcObject = stream;
            video.play();
        };

        if (navigator.mozGetUserMedia) {
            navigator.mozGetUserMedia(videoObj, getStream, errBack);
        }
    }
</script>
</body>

```

Evento ativado quando o parsing da página termina

Função que é chamada em caso de erro no vídeo

Função que é chamada com o stream de vídeo a processar

Objeto que define os tipos de media pedidos e os requisitos. Ex: { audio: true, video: { width: 1280, height: 720 } }

Multimédia: *Player* de Vídeo



Player de Vídeo: Vídeo

```
<head>
    <script src="js/VideoPlayer_manager1.js"></script>
</head>
<body>
<header>
    <h1>Video Player</h1>
</header>

<section id="playerVideo">
    <video>
        <source src="video/13_MissionImpossible_KyleCooper.ogv"
type="video/ogg">
    </video>

    <div id="controls">
        <button id="playpause" alt="play" title="play"
onclick="togglePlay()">play</button>
    </div>
</section>

<section id='start'>
    <script>window.onload = new_videoPlayer();</script>
</section>
```

Player de Vídeo: Botões “Pause/Play”

```
function new_videoPlayer() {
    video = document.getElementsByTagName("video")[0];
    video.controls = false;
    let ppbutton = document.getElementById("playpause");

    video.addEventListener('play', function () {
        ppbutton.title = "pause";
        ppbutton.innerHTML = "pause";
    }, false);

    video.addEventListener('pause', function () {
        ppbutton.title = "play";
        ppbutton.innerHTML = "play";
    }, false);
}

function togglePlay() {
    if (video.paused || video.ended) {
        if (video.ended) video.currentTime = 0;
        video.play();
    }
    else {
        video.pause();
    }
}
```

(1) Após o clique no botão “playpause” é executada a função togglePlay.

(2) A função togglePlay faz “play” ou “pause” que geram automaticamente os eventos ‘play’ and ‘pause’.

Player de Vídeo: “Botão Stop”

```
<div id="controls">
  <button id= "playpause" alt="play" title="play"
  onclick="togglePlay()">play</button>

  <button id= "stop" alt="stop" title="stop"
  onclick="stopVideo()">stop</button>
</div>
```

...

```
video.addEventListener('ended', function() { this.pause(); }, false);

function stopVideo() {
  video.pause();
  video.currentTime = 0;
}
```



Player de Vídeo: Botões “rewind” e “forward”

```
<button id="rewind" alt="rewind" title="rewind"
        onclick="changePlaybackSpeed('>');">&laquo;</button>

<button id="ffwd" alt="fast forward" title="fast forward"
        onclick="changePlaybackSpeed('+>');">&raquo;</button>

function changePlaybackSpeed(direction) {
    if (video.playbackRate != undefined) {
        if (direction == "-") video.playbackRate -= 1;
        else video.playbackRate += 1;
    }
    else {
        if (direction == "-") video.currentTime -= 1;
        else video.currentTime += 1;
    }
}
```

Player de Vídeo: Volume

```
<button id="volumeDown" alt="decrease volume" title="-"
onclick="changeVolume('−')"></button>
<button id="volumeUp" alt="increase volume" title="+"
onclick="changeVolume('+')"></button>

function changeVolume(direction) {
    let volume = Math.floor(video.volume * 10) / 10;
    video.muted = false;
    if (direction == "−") {
        if (volume <= 0.1) video.volume = 0;
        else video.volume -= 0.1;
    }
    else {
        if (volume >= 0.9) video.volume = 1;
        else video.volume += 0.1;
    }
}
```

Player de Vídeo: Botão “mute”

```
<button id="mute" alt="mute" title="mute"
    onclick="toggleMute()">mute</button>

function toggleMute() {
    let mute = document.getElementById("mute");
    if (video.muted) {
        mute.innerHTML = "mute";
        video.muted = false;
    }
    else {
        mute.innerHTML = "unmute";
        video.muted = true;
    }
}
```



Player de Vídeo: Barra de Progresso

```
<style>
    #progressBar { border:1px solid #aaa; color:#fff; width:292px; height:
                    20px; }
    #played { background-color:#aaa; height:20px; display:inline-block; }
</style>
...
<div id="controls">
    <div id="progressBar"><span id="played"></span></div>
    ...
</div>
...
video.addEventListener('timeupdate', function() {
    let value = 0;
    if (video.currentTime > 0) {
        value = Math.floor((100 / video.duration) * video.currentTime);
    }
    document.getElementById("played").style.width = value + "%";
}, false);
```

Evento ocorre quando é alterado o instante de tempo que está a ser reproduzido do áudio/vídeo

Player de Vídeo: Click na Barra de Progresso

```
let progressBar =  
document.getElementById("progressBar").addEventListener("mouseup",  
    function(e) { setPlayPosition(e.pageX); }, false);  
  
function setPlayPosition(x) {  
    let progressBar = document.getElementById("progressBar");  
    let value = (x - findPos(progressBar)).toFixed(2);  
    let timeToSet = ((video.duration / progressBar.offsetWidth).toFixed(2) *  
        value).toFixed(2);  
    video.currentTime = timeToSet;  
}  
  
function findPos(element) {  
    let top = 0;  
    let left = 0;  
    let offsetParent = element.offsetParent;  
    while (offsetParent != null) {  
        top += offsetParent.offsetTop;  
        left += offsetParent.offsetLeft;  
        offsetParent = offsetParent.offsetParent;  
    }  
    return {top: top, left: left};  
}
```

Função para encontrar a posição real do elemento

Retorna a largura do elemento incluindo “border”, “padding” e “scrollbar”

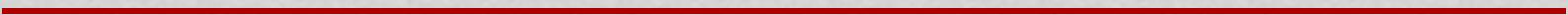
Player de Vídeo: findPos()

```
function findPos(obj) {  
    let curleft = 0;  
    if (obj.offsetParent) {  
        do {  
            curleft += obj.offsetLeft; ←  
            obj = obj.offsetParent;  
        } while (obj);  
    }  
    return curleft;  
}
```

Retorna o primeiro elemento *parent* posicionado (propriedade *position* diferente de “static”)
Retorna “null” se o elemento tem a propriedade *display* igual a “none”

Retorna a posição (offset) horizontal do elemento

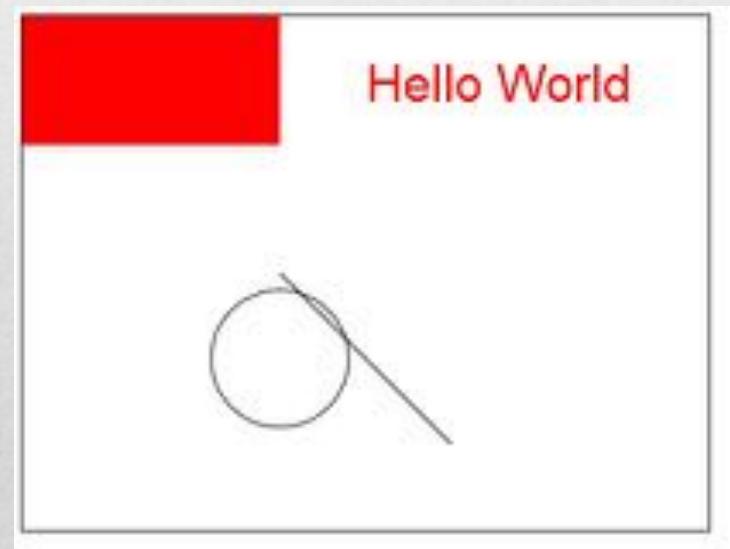
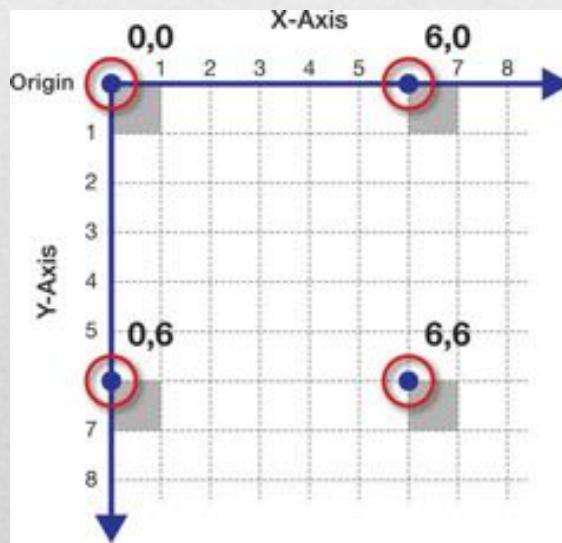
Canvas



Multimédia: Canvas

■ Elemento <canvas>

- É criada uma área rectangular na página onde se pode adicionar linhas, gráficos, texto, imagens, vídeos...
- Origem do elemento <canvas> no canto superior esquerdo



Multimédia : Canvas - Exemplo

```
<body>
  <canvas id="myCanvas" width="400" height="300"> </canvas>
  <script>
    let c = document.getElementById("myCanvas");
    let ctx = c.getContext("2d");

    ctx.fillStyle="#FF0000";
    ctx.fillRect(0,0,150,75);

    ctx.moveTo(150,150);
    ctx.lineTo(250,250);
    ctx.stroke();

    ctx.beginPath();
    ctx.arc(150,200,40,0,2*Math.PI);
    ctx.stroke();

    ctx.font="30px Arial";
    ctx.fillText("Hello World",200,50);
  </script>
</body>
```

Objeto para desenhar em 2D
(Também podia ser 3D)

Método para definir o estilo da cor de preenchimento

Método para desenhar retângulo preenchido com a cor

Método para desenhar uma linha

Método para desenhar círculos e arcos

Método para desenhar texto preenchido com a cor

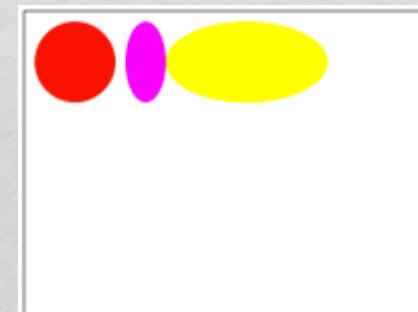
Multimédia : Canvas – Desenhar Círculos/Elipses

```
...
<body>
<canvas id="myCanvas" width="400" height="300"></canvas>
<script>
    let c = document.getElementById("myCanvas");
    let ctx = c.getContext("2d");

    ctx.save();
    ctx.translate(150,150);
    ctx.scale(2,1);
    ctx.fillStyle = "green";
    ctx.beginPath();
    ctx.arc(0, 0, 40, 0, 2*Math.PI,true);
    ctx.closePath();
    ctx.fill();
    ctx.restore();
</script>

</body>
```

Define se o arco deve ser desenhado no sentido contrário aos ponteiros do relógio. Por omissão, está definido como “false”, isto é, no sentido dos ponteiros do relógio.



Multimédia : Canvas – Desenhar Imagens

```
<body>
<canvas id="myCanvas" width="400" height="300" ></canvas>
<script>
    let c = document.getElementById("myCanvas");
    let ctx = c.getContext("2d");

    let im = new Image();
    im.onload = function () {
        ctx.drawImage(im, 100, 100, 100, 100);
    }
    im.src = "images/annika.jpg"

</script>
</body>
```

Função para desenhar no *canvas* uma imagem

Utiliza-se o evento **onload** para garantir que a imagem só possa ser processada após ter sido carregada em memória.

Multimédia : Canvas – Coração

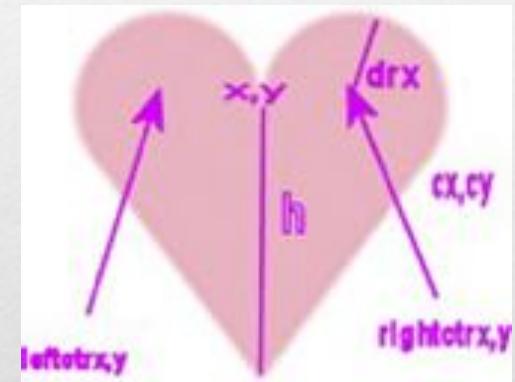
```

<canvas id="myCanvas" width="400" height="300" ></canvas>
<script>
    let c = document.getElementById("myCanvas");
    let ctx = c.getContext("2d");

    function Heart(x,y,w,c) {
        let h = w * 0.7;
        let drx = w/4;
        let ang = .25 * Math.PI;
        let leftctrx = x - drx;
        let rightctrx = x + drx;
        let cx = rightctrx + drx * Math.cos(ang);
        let cy = y + drx * Math.sin(ang);

        ctx.fillStyle = c;
        ctx.beginPath();
        ctx.moveTo(x, y);
        ctx.arc(leftctrx, y, drx, 0, Math.PI - ang, true);
        //ctx.arc(leftctrx, y, drx, Math.PI - ang, 2*Math.PI);
        ctx.lineTo(x, y + h);
        ctx.lineTo(cx, cy);
        ctx.arc(rightctrx, y, drx, ang, Math.PI, true);
        //ctx.arc(rightctrx, y, drx, Math.PI, 2*Math.PI+ang);
        ctx.closePath();
        ctx.fill();
    }
    Heart(100,100,100,"pink");
</script>

```



Multimédia: Canvas – Propriedades

Propriedades	Descrição
strokeStyle	Define ou retorna a cor, o gradiente ou o padrão usado nas linhas
shadowColor	Define ou retorna a cor usada para as sombras
shadowBlur	Define ou retorna o nível de <i>blur</i> para as sombras
shadowOffsetX	Define ou retorna a distância horizontal da sombra ao objeto
shadowOffsetY	Define ou retorna a distância vertical da sombra ao objeto
linecap	Define ou retorna o estilo das extremidades da linha
lineJoin	Define ou retorna o tipo de canto criado quando 2 linhas se juntam
lineWidth	Define ou retorna a largura da linha atual
textAlign	Define ou retorna o alinhamento atual do texto
textBaseline	Define ou retorna o alinhamento vertical do texto em relação à linha de referência
globalAlpha	Define ou retorna a transparência (canal alpha) atual dos objetos

Multimédia: Canvas – Métodos

Métodos	Descrição
createLinearGradient	Criar um gradiente linear a utilizar em conteúdo no canvas
rect	Cria um retângulo
strokeRect	Desenha um retângulo (sem ser preenchido com cor)
clearRect	Apaga uns pixeis específico do canvas relativos a um retângulo
clip	Seleciona/corta uma região do canvas
arcTo	Cria um arco/curva entre duas tangentes
rotate	Roda o objeto atual
strokeText	Desenha texto (sem estar preenchido)
createPattern	Repete um elemento específico numa direção
quadraticCurveTo	Cria uma curva quadratica Bézier